CVE-2010-4221

Carleigh Duncan[1], Venezia Cruz[2]
Computer Science Department, Towson University
8000 York Road, Towson, MD 21252, United States of America
[1]cdunca8@students.towson.edu
[2]vcruz2@students.towson.edu

***Abstract* - Using a 32-bit Ubuntu 10.04 version virtual machine with 2.6.35-19-generic kernel, an exploit of the ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow vulnerability was attempted using Metasploit on a Kali Linux virtual machine.**

***Keywords* - CVE-2010-4221, ProFTPD, vulnerability, Metasploit, execute code overflow**

I. INTRODUCTION

Ubuntu is the the most popular operating system for the cloud, and the referenced operating system for OpenStack [1]. It is an open-source distribution of Linux that is based on Debian [2]. Ubuntu, as well as many other operating systems, employ ProFTPD as a server. This allows a system to transfer files between systems (a client and a server).

II. VULNERABILITY

CVE-2010-4221 is a Linux-specific vulnerability affecting ProFTPD versions 1.2.8 to 1.3.3b, that is, FTP or FTPS servers. Through it, an attacker may completely compromise the system's integrity, view all system files, or even shut down the target; all of this without particular access conditions or authentication.

A. ProFTPD

There exists multiple stack-based buffer overflows in the pr_netio_telnet_gets() function in the netio.c file of ProFTPD in versions prior to 1.3.3c [3]. The buffer length value *buflen* did not encounter a check to confirm its value was greater than zero. If a value less than or equal to zero was passed through the switch statement, shown in Fig. 1 (excluding the two lines in red), the value of *buflen* would inevitably be decremented twice. This would cause buffer overflow. If the value of TELNET_IAC was somehow set to 1, this would also cause the overflow.

B. Canary Word

Canary words are known values that are placed on the stack, between a buffer and control data, to monitor for buffer overflows. If the value of the canary word was known, it could be used to either infiltrate the kernel code, or inject malicious code. In ProFTPD, the canary word has 24 bits of entropy, but the canary word does not reset after failed attempts to guess it [3]. An attacker could continue guessing the cookie value and eventually exploit the vulnerability through brute force.

Although Linux distributions do not ship vulnerable operating system versions, either due to fixing the vulnerability or due to the inclusion of stack smashing protection, stack smashing protection will not prevent exploitation [4].

III. EXPLOIT

To exploit this vulnerability, the attacker makes use of the multiple possibilities for buffer overflow in the pr_netio_telnet_gets() function. Data containing an ample number of Telnet IAC commands are used to corrupt the memory enabling the attacker to execute their own code with impunity.

Through our research, we found that our exploitation attempt from an average home system may take several days. This is a semi-reasonable time frame.

This vulnerability has an available Metasploit exploit module.

A. Explanation

The exploit begins with establishing the target and intent. In the Metasploit module that was used, this means establishing that the IACCount be 8192; large enough to crash the target by writing off the end of the stack [5].

Alternatively, an attacker can load their own code as a writable by quadruple dereferencing the 'res' pointer, skipping the pool chunk header and then jumping to their own data. This is the Debian Squeeze version of the exploit, and is not what was attempted.

With the intent there, the exploit proceeds to use a banner to check against the target. The exploit needs to be sure that the target is running a vulnerable version. If it is, a canary word will be needed to manipulate the buffer. This value doesn't change while proftpd forks children, and thus can be attempted forever. A caveat that exists is that instability cannot be induced. There are bad characters that could have been introduced to addresses previously, which are defined along with the intent. If the canary word happens to contain one of the defined bad characters, the exploit cannot proceed.

The exploit tries to create sessions with randomly generated canary words until something sticks.

When the canary is guessed, the exploit returns to the buffer-data boundary, checks for the value, and overwrites with impunity. For cleanup, the stack may need to be polished by giving ProFTPD new values for ebx, esi and ebp, but this was skipped in this attempt: total shutdown was the goal.

IV. IMPACT

A. Confidentiality

There is a complete breach of the system. All files become accessible in all ways, which inherently breaks any confidentiality or privacy of the files on the ProFTPD-using host. In short, there is total disclosure of the target's possessions. More practically, since the attacker also knows the canary value, they could walk through the buffer to read whatever other values were previously stored.

B. *Integrity*

Because there is a complete breach, the attacker should also be able to overwrite or otherwise compromise any host files. There is a complete loss of security here, and as such, the attacked can gain full permissions. Worse, because this vulnerability allows everything up to the running of arbitrary code, these permissions can be extended to the attacker's discretion.

C. *Availability*

It is possible to cause a disruption of service condition, even if the exploit attempt fails. As a result, availability is compromised by this exploited vulnerability.

V. PATCH/UPDATE

A. *ProFTPD Buffer Overflow Patch*

The code for the ProFTPD was updated for all versions following 1.3.3b with two lines of code, shown in red in Fig. 1. The purpose of the two lines of code is to check the value of *buflen* and make sure the value is not 0; if the value is 0, a break is issued in order to avoid a decrement function that would cause the stack to overflow [6].

VI. EXPLOIT EXPERIENCE

The exploit was initiated from an up-to-date Kali Linux virtual machine (VM, or box) through Metasploit, using the "ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow" exploit. The target chosen was the "ProFTPD 1.3.2c Server (Ubuntu 10.04)". Once the RHOST and RPORT was set to the correct settings (in this case, the victim Ubuntu box had an open port 22, and had an IP address of 192.168.32.133), the exploit was run. The exploit initially requires the brute forcing of the 24-bit canary word, which could take multiple days to perform.

A. *Issues*

After running the exploit for 28 hours, and achieving 77% of all possible canary values, we came to the realization that the Ubuntu box did not have the correct version of ProFTPD installed. As a result, even if the canary value was brute-forced, the exploit may fail.

The creation of an appropriate Ubuntu VM was attempted, but also ran into multiple problems. The correct version of Ubuntu was 10.04, also known as Lucid Lynx. This version is no longer supported with viable updates, so attempts to use terminal commands to install updates (such as for OpenSSH and ProFTPD) failed. Attempts to manually download and install the correct versions also failed, as there were multiple directories and dependencies that no longer exist on the internet that were required for successful installation. It is still possible that a private user or businesses could still be running on vulnerable versions; we have no knowledge of them.

Another issue encountered in the second Ubuntu box involved the ports. The Ubuntu 10.04 VM had no applications listening to any ports and we could not fix this issue, and as a result, no ports could be kept open.

VII. CONCLUSION

The ProFTPD vulnerability described is an extremely dangerous vulnerability for the systems that are affected by it. It has a CVSS score of 10, as it allows for complete violation of the CIA triad without any need to authenticate in order for the exploit to function.

That being said, the systems impacted by this vulnerability are few and far between: the only systems impacted by this vulnerability are Linux machines running a version of ProFTPD between 1.2.8 to 1.3.3b. These have long since been patched and updated, as most ProFTPD is now running on an updated version of 1.3.6.

This vulnerability does show how easy it is to allow for stack and buffer overflow, as one simple check function is all that was required to mitigate this exploitable vulnerability, as shown in Figure 1.

REFERENCES

[1] "Public cloud | Ubuntu", *Ubuntu.com*, 2018. [Online]. Available: https://www.ubuntu.com/public-cloud. [Accessed: 26- Nov- 2018].
[2]"Debian | Ubuntu", *Ubuntu.com*, 2018. [Online]. Available: https://www.ubuntu.com/community/debian. [Accessed: 26- Nov- 2018].
[3]"CVE-2010-4221 ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux) | Rapid7", *Rapid7.com*, 2018. [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/ftp/proftp_telnet_iac. [Accessed: 26- Nov- 2018].
[4]"CVE-2010-4221 : Multiple stack-based buffer overflows in the pr_netio_telnet_gets function in netio.c in ProFTPD before 1.3.3c allow rem", *Cvedetails.com*, 2018. [Online]. Available: https://www.cvedetails.com/cve/cve-2010-4221. [Accessed: 25- Nov- 2018].
[5]"rapid7/metasploit-framework", *GitHub*, 2018. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/linux/ftp/proftp_telnet_iac.rb. [Accessed: 29- Nov- 2018].
[6]A. Pingios, "CVE-2010-4221: ProFTPd TELNET_IAC Remote Stack Overflow", *xorl %eax, %eax*, 2018. [Online]. Available: https://xorl.wordpress.com/2010/11/15/cve-2010-4221-proftpd-telnet_iac-remote-stack-overflow/. [Accessed: 25- Nov- 2018].

```
#define PR_DEFAULT_CMD_BUFSZ
          (PR_TUNABLE_PATH_MAX + 7)
   ...
int pr_cmd_read(cmd_rec **res) {
  static long cmd_bufsz = -1;
  char buf[PR_DEFAULT_CMD_BUFSZ+1] = {'\0'};
  char *cp;
  size_t buflen;
   ...
  while (TRUE) {
    pr_signals_handle();
    memset(buf, '\0', sizeof(buf));

    if (pr_netio_telnet_gets(buf, sizeof(buf)-1,
session.c->instrm,
        session.c->outstrm) == NULL) {
   ...
  }
  return 0;
}


char *pr_netio_telnet_gets(char *buf, size_t buflen,
pr_netio_stream_t *in_nstrm, pr_netio_stream_t
*out_nstrm)
{
  char *bp = buf;
  unsigned char cp;
  int toread, handle_iac = TRUE, saw_newline =
FALSE;
  pr_buffer_t *pbuf = NULL;

  if (buflen == 0) {
          errno = EINVAL;
          return NULL;
  }
          ...
  buflen--;
```

```
  if (in_nstrm->strm_buf)
          pbuf = in_nstrm->strm_buf;
  else
          pbuf = netio_buffer_alloc(in_nstrm);
  while (buflen) {
          ...
          while (buflen && toread > 0 && *pbuf-
>current != '\n' && toread--) {
          cp = *pbuf->current++;
          pbuf->remaining++;
          if (handle_iac == TRUE) {
          switch (telnet_mode) {
          case TELNET_IAC:
          switch (cp) {
          case TELNET_WILL:
          case TELNET_WONT:
          case TELNET_DO:
          case TELNET_DONT:
          case TELNET_IP:
          case TELNET_DM:
          ...
          default:
                    *bp++ = TELNET_IAC;
                    buflen--;
                    telnet_mode = 0;
                    break;
          }
          …
           if(buflen == 0)
           break;
          *bp++ = cp;
          buflen--;
          }
          ...
  properly_terminated_prev_command = TRUE;
  *bp = '\0';
  return buf;
}
```

Fig. 1 Edited version of the code responsible for the ProFTPD buffer overflow vulnerability. The code in red was added to fix the vulnerability.